Entity Resolution:
Glue for Middleware

Hector Garcia-Molina
Stanford University
Middleware

apps

middleware

System 1

⋯

System n
Middleware

apps

middleware

System 1

... System n

what matches what??
Matching

• Execution Level
  – matching ports, calls, parameters, workflows ...

• Data Level
  – matching records, attributes, values ...
Matching

• Execution Level

• Data Level
  – Ontology
  – Schema
  – Instance
Example: Stock Options

- Ontology
  - Strike Price
  - Stock Option
  - Stock Grant
  - Date of Option
  - Taxable Income
  - Black–Scholes Valuation
  - Market Valuation
  - Deferred Compensation
Example: Stock Options

- Schema

  Stock_Option(Date, Price, Shares, Holder, Plan, ...)

  Option(Date, StrikePrice, Shares, Employee, Restrictions, ...)
Example: Stock Options

• Instance

Option 1
- Name: Tom S. Smith
- Adr: 123 Main St
- Date:
- Shares:

Option 2
- Name: Thomas Smith
- Adr: 132 Main St
- Date:
- Shares:
This Talk

• Instance Resolution
  – a.k.a. Entity Resolution
  – a.k.a. De-Duplication
  – a.k.a. Record Linkage
Applications

- comparison shopping
- mailing lists
- classified ads
- customer files
- counter-terrorism
Why is ER Challenging?

- Huge data sets
- No unique identifiers
- Lots of uncertainty
- Many ways to skin the cat
Outline

• Taxonomy
• Swoosh Algorithm
• Distributed ER
• More on blocking
Taxonomy: Pairwise vs Global

- Decide if $r$, $s$ match only by looking at $r$, $s$?
- Or need to consider more (all) records?

```
Nm: Pat Smith
   Ad: 123 Main St
   Ph: (650) 555-1212

Nm: Patrick Smith
   Ad: 132 Main St
   Ph: (650) 555-1212

Nm: Patricia Smith
   Ad: 123 Main St
   Ph: (650) 777-1111
```
Taxonomy: Pairwise vs Global

• Global matching complicates things a lot!
  – e.g., change decision as new records arrive

Nm: Pat Smith
  Ad: 123 Main St
  Ph: (650) 555-1212

or

Nm: Patrick Smith
  Ad: 132 Main St
  Ph: (650) 555-1212

Nm: Patricia Smith
  Ad: 123 Main St
  Ph: (650) 777-1111
Taxonomy: Outcome

• Partition of records
  – e.g., comparison shopping

• Merged records

Nm: Pat Smith
- Ad: 123 Main St
- Ph: (650) 555-1212

Nm: Patricia Smith
- Ad: 123 Main St
- Ph: (650) 555-1212

Nm: Patricia Smith
- Ad: 132 Main St
- Ph: (650) 777-1111
- Hair: Black
Taxonomy: Outcome

• Iterate after merging

Nm: Tom
  Ad: 123 Main
  BD: Jan 1, 85
  Wk: IBM
  Oc: lawyer

Nm: Thomas
  Ad: 123 Maim
  Oc: lawyer

Nm: Tom
  Wk: IBM
  Oc: laywer
  Sal: 500K

Nm: Tom
  Ad: 123 Main
  BD: Jan 1, 85
  Wk: IBM
  Oc: lawyer

Nm: Tom
  Ad: 123 Main
  BD: Jan 1, 85
  Wk: IBM
  Oc: lawyer
  Sal: 500K
Taxonomy: Record Reuse

• One record related to multiple entities?

Nm: Pat Smith Sr.
  Ph: (650) 555-1212

Nm: Pat Smith Sr.
  Ph: (650) 555-1212
  Ad: 123 Main St

Nm: Pat Smith Jr.
  Ph: (650) 555-1212

Nm: Pat Smith Jr.
  Ph: (650) 555-1212
  Ad: 123 Main St
Taxonomy: Record Reuse

- Partitions
  - $r$
  - $s$
  - $t$

- Merges
  - $r$
  - $s$
  - $t$
  - $rs$
  - $st$
Taxonomy: Record Reuse

• Partitions

• Merges

• Record reuse $\rightarrow$ complex and expensive!
Taxonomy: Multiple Entity Types

- person 1
  - brother
  - member

- person 2
  - member

- Organization A
  - business

- Organization B
Taxonomy: Multiple Entity Types

authors

papers

same??
Taxonomy: Exact vs Approximate

products

- cameras
- CDs
- books

...
Taxonomy: Exact vs Approximate

terrorists
\[ \text{sort by age} \]
terrorists

B Cooper 30

\{ match against ages 25-35 \}
Taxonomy: Other Variations

• Managing uncertainty
• Similarity computation
Outline

• Taxonomy
• Swoosh Algorithm
• Distributed ER
• More on blocking
Scenario

• Pairwise matching
• Record merging
• No record reuse
• Single entity type
Model

\[ M(r_1, r_2) \]

\[ M(r_4, r_3) \]

\[ r_4: \langle r_1, r_2 \rangle \]

\[ \langle r_4, r_3 \rangle \]
Correct Answer

ER(R) = All derivable records.....
Minus “dominated” records
Question

• What is best sequence of match, merge calls that give us right answer?
Brute Force Algorithm

• Input R:
  – r1 = [a:1, b:2]
  – r2 = [a:1, c: 4, e:5]
  – r3 = [b:2, c:4, f:6]
  – r4 = [a:7, e:5, f:6]
Brute Force Algorithm

• Input R:
  – r1 = [a:1, b:2]
  – r2 = [a:1, c: 4, e:5]
  – r3 = [b:2, c:4, f:6]
  – r4 = [a:7, e:5, f:6]

• Match all pairs:
  – r1 = [a:1, b:2]
  – r2 = [a:1, c: 4, e:5]
  – r3 = [b:2, c:4, f:6]
  – r4 = [a:7, e:5, f:6]
  – r12 = [a:1, b:2, c:4, e:5]
Brute Force Algorithm

• Match all pairs:
  – r1 = [a:1, b:2]
  – r2 = [a:1, c: 4, e:5]
  – r3 = [b:2, c:4, f:6]
  – r4 = [a:7, e:5, f:6]
  – r12 = [a:1, b:2, c:4, e:5]

• Repeat:
  – r1 = [a:1, b:2]
  – r2 = [a:1, c: 4, e:5]
  – r3 = [b:2, c:4, f:6]
  – r4 = [a:7, e:5, f:6]
  – r12 = [a:1, b:2, c:4, e:5]
  – r123 = [a:1, b:2, c:4, e:5, f:6]
Question # 1

Brute Force Algorithm

• Input R:
  – r1 = [a:1, b:2]
  – r2 = [a:1, c: 4, e:5]
  – r3 = [b:2, c:4, f:6]
  – r4 = [a:7, e:5, f:6]

• Match all pairs:
  – r1 = [a:1, b:2]
  – r2 = [a:1, c: 4, e:5]
  – r3 = [b:2, c:4, f:6]
  – r4 = [a:7, e:5, f:6]
  – r12 = [a:1, b:2, c:4, e:5]

Can we delete r1, r2?
Brute Force Algorithm

• Match all pairs:
  - r1 = [a:1, b:2]
  - r2 = [a:1, c:4, e:5]
  - r3 = [b:2, c:4, f:6]
  - r4 = [a:7, e:5, f:6]
  - r12 = [a:1, b:2, c:4, e:5]

• Repeat:
  - r1 = [a:1, b:2]
  - r2 = [a:1, c:4, e:5]
  - r3 = [b:2, c:4, f:6]
  - r4 = [a:7, e:5, f:6]
  - r12 = [a:1, b:2, c:4, e:5]
  - r123 = [a:1, b:2, c:4, e:5, f:6]

Can we avoid comparisons?
ICAR Properties

• Idempotence:
  – $M(r_1, r_1) = true; <r_1, r_1> = r_1$

• Commutativity:
  – $M(r_1, r_2) = M(r_2, r_1)$
  – $<r_1, r_2> = <r_2, r_1>$

• Associativitiy
  – $<r_1, <r_2, r_3>> = <<r_1, r_2>, r_3>$
More Properties

• Representativity
  – If \( <r_1, r_2> = r_3 \), then for any \( r_4 \) such that \( M(r_1, r_4) \) is true we also have \( M(r_3, r_4) = \text{true} \).
ICAR Properties ➔ Efficiency

- Commutativity
- Idempotence
- Associativity
- Representativity

- Can discard records
- ER result independent of processing order
Swoosh Algorithms

• Record Swoosh
  • Merges records as soon as they match
  • Optimal in terms of record comparisons

• Feature Swoosh
  • Remembers values seen for each feature
  • Avoids redundant value comparisons
Swoosh Performance
If ICAR Properties Do Not Hold?

r1: [Joe Sr., 123 Main, Ph: 123, DL:X]

r2: [Joe, 123 Main, Ph: 123]

r3: [Joe Jr., 123 Main, DL:Y]

r12: [Joe Sr., 123 Main, Ph: 123, DL:X]

r23: [Joe Jr., 123 Main, Ph: 123, DL:Y]
If ICAR Properties Do Not Hold?

Full Answer: $ER(R) = \{r12, r23, r1, r2, r3\}$
Minus Dominated: $ER(R) = \{r12, r23\}$
If ICAR Properties Do Not Hold?

Full Answer: \( ER(R) = \{r12, r23, r1, r2, r3\} \)

Minus Dominated: \( ER(R) = \{r12, r23\} \)

R-Swoosh Yields: \( ER(R) = \{r12, r3\} \) or \( \{r1, r23\} \)
Swoosh Without ICAR Properties

![Graph showing result size vs. title threshold](image-url)
Distributed Swoosh

P1  P2  P3

r1  r2  r3  r4  r5  r6  ...

...
Distributed Swoosh

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>r1</td>
<td>r2</td>
</tr>
<tr>
<td>r3</td>
<td>r2</td>
<td>r3</td>
</tr>
<tr>
<td>r4</td>
<td>r4</td>
<td>r5</td>
</tr>
<tr>
<td>r5</td>
<td>r5</td>
<td>r6</td>
</tr>
<tr>
<td>r6</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
DSwoosh Performance

![Graph showing performance comparison between Sequential, Grid, Linear Ordering, and Value Equality methods.](image-url)
Outline

• Swoosh Algorithm
• Distributed ER
• More on blocking
Iterative Blocking: Example

<table>
<thead>
<tr>
<th>Record</th>
<th>Name</th>
<th>Addr (zip)</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>John Doe</td>
<td>52139</td>
<td>jdoe@yahoo</td>
</tr>
<tr>
<td>s</td>
<td>John Doe</td>
<td>94305</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>J. Foe</td>
<td>94305</td>
<td>jdoe@yahoo</td>
</tr>
<tr>
<td>u</td>
<td>Bobbie Brown</td>
<td>12345</td>
<td>bob@gmail</td>
</tr>
<tr>
<td>v</td>
<td>Bobbie Brown</td>
<td>12345</td>
<td>bob@gmail</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Record</th>
<th>Name</th>
<th>Addr (zip)</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>John Doe</td>
<td>52139</td>
<td>jdoe@yahoo</td>
</tr>
<tr>
<td>s</td>
<td>John Doe</td>
<td>94305</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>J. Foe</td>
<td>94305</td>
<td>jdoe@yahoo</td>
</tr>
<tr>
<td>u</td>
<td>Bobbie Brown</td>
<td>12345</td>
<td>bob@gmail</td>
</tr>
<tr>
<td>v</td>
<td>Bobbie Brown</td>
<td>12345</td>
<td>bob@gmail</td>
</tr>
</tbody>
</table>

Iterative ER:

\(<r, s, t>\)

\(<r, s>\)

(John Doe, \{52139, 94305\}, jdoe@yahoo)
## Blocking

<table>
<thead>
<tr>
<th>Record</th>
<th>Name</th>
<th>Addr (zip)</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>John Doe</td>
<td>52139</td>
<td>jdoe@yahoo</td>
</tr>
<tr>
<td>s</td>
<td>John Doe</td>
<td>94305</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>J. Foe</td>
<td>94305</td>
<td>jdoe@yahoo</td>
</tr>
<tr>
<td>u</td>
<td>Bobbie Brown</td>
<td>12345</td>
<td>bob@gmail</td>
</tr>
<tr>
<td>v</td>
<td>Bobbie Brown</td>
<td>12345</td>
<td>bob@gmail</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Partition by</th>
<th>$b_{-,1}$</th>
<th>$b_{-,2}$</th>
<th>$b_{-,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC1</td>
<td>zip code</td>
<td>r</td>
<td>s, t</td>
<td>u, v</td>
</tr>
<tr>
<td>SC2</td>
<td>1st char last name</td>
<td>r, s</td>
<td>t</td>
<td>u, v</td>
</tr>
</tbody>
</table>
Blocking

<table>
<thead>
<tr>
<th>Record</th>
<th>Name</th>
<th>Addr (zip)</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>John Doe</td>
<td>52139</td>
<td>jdoe@yahoo</td>
</tr>
<tr>
<td>s</td>
<td>John Doe</td>
<td>94305</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>J. Foe</td>
<td>94305</td>
<td>jdoe@yahoo</td>
</tr>
<tr>
<td>u</td>
<td>Bobbie Brown</td>
<td>12345</td>
<td>bob@gmail</td>
</tr>
<tr>
<td>v</td>
<td>Bobbie Brown</td>
<td>12345</td>
<td>bob@gmail</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Partition by</th>
<th>b,1</th>
<th>b,2</th>
<th>b,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC1</td>
<td>zip code</td>
<td>r</td>
<td>s, t</td>
<td>u,v</td>
</tr>
<tr>
<td>SC2</td>
<td>1st char last name</td>
<td>r, s</td>
<td>t</td>
<td>u, v</td>
</tr>
</tbody>
</table>

Will miss: < r, s, t >
## Blocking

<table>
<thead>
<tr>
<th>Record</th>
<th>Name</th>
<th>Addr (zip)</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>John Doe</td>
<td>52139</td>
<td>jdoe@yahoo</td>
</tr>
<tr>
<td>s</td>
<td>John Doe</td>
<td>94305</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>J. Foe</td>
<td>94305</td>
<td>jdoe@yahoo</td>
</tr>
<tr>
<td>u</td>
<td>Bobbie Brown</td>
<td>12345</td>
<td>bob@gmail</td>
</tr>
<tr>
<td>v</td>
<td>Bobbie Brown</td>
<td>12345</td>
<td>bob@gmail</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Partition by</th>
<th>b_{-,1}</th>
<th>b_{-,2}</th>
<th>b_{-,3}</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC1</td>
<td>zip code</td>
<td>r</td>
<td>s, t</td>
<td>u, v</td>
</tr>
<tr>
<td>SC2</td>
<td>1st char last name</td>
<td>r, s</td>
<td>t</td>
<td>u, v</td>
</tr>
</tbody>
</table>

Solution: Propagate Matches
What We Have Done

• Formal Model for Iterative Blocking
  – based on generic “core” ER algorithm

• Two Algorithms:
  – Lego: in memory
  – Duplo: disk based

• Experiments
Sample Results: Accuracy
Sample Results: Run Time
Conclusion

• ER is old and important problem
• Critical for gluing components
Thanks.